Library functions

Table 1 gives a list of library functions. Here the arguments `c` denotes a character, `d` a double precision, `f` a file, `i` an integer, `l` a long integer, `p` a pointer, `s` a string and `u` an unsigned integer. The exponential base is the *natural number*, e which is equal to 2.7182818.... Arguments for the functions `printf`, `scanf`, `fprintf` and `fscanf` are here left out to be found elsewhere.

Function	*Type*	*Description*	*Header*
`abs(i)`	int	absolute value	`stdlib`
`acos(d)`	double	arc cosine	`math`
`asin(d)`	double	arc sine	`math`
`atan(d)`	double	arc tangent	`math`
`atan2(d1,d2)`	double	arc tangent of d_1/d_2	`math`
`atof(s)`	double	convert string to double precision	`stdlib`
`atoi(s)`	int	convert string to an integer	`stdlib`
`atol(s)`	long	convert string to a long integer	`stdlib`
`calloc(u1,u2)`	void*	allocate an array of u_1 elements, each of length u_2 bytes, return a pointer to the beginning of the space allocated	`stdlib`
`ceil(d)`	double	round up to the next higher integer	`math`
`cos(d)`	double	cosine	`math`
`cosh(d)`	double	hyperbolic cosine	`math`
`difftime(l1,l2)`	double	time difference $l_1 - l_2$	`time`
`exit(u)`	void	close all files and buffers, terminate the programme with termination status `u`	`stdlib`
`exp(d)`	double	exponential e^d	`math`
`fabs(d)`	double	absolute value	`math`
`fclose(f)`	int	close file, return 0 if successful	`stdio`
`feof(f)`	int	test whether end of file, return 0 if unsuccessful and a nonzero otherwise	`stdio`
`fgetc(f)`	int	read a character from a file	`stdio`
`fgets(s,i,f)`	char*	read string `s`, containing `i` characters, from a file `f`	`stdio`
`floor(d)`	double	round down to the next lower integer	`math`
`fmod(d1,d2)`	double	the remainder of d_1/d_2, with sign the same as d_1	`math`
`fopen(s1,s2)`	file*	open a file f_1 of type f_2, return a pointer to the file	`stdio`
`fprintf(f,...)`	int	write data to a file	`stdio`
`fputc(c,f)`	int	write a character to a file	`stdio`
`fputs(s,f)`	int	write a sting `s` to file `f`	`stdio`
`fread(s,i1,i2,f)`	int	read from file `f` to a string `s` i_2 data items, each of i_1 bytes	`stdio`
`free(p)`	void	free a block of memory the beginning of which is `p`	`malloc`, `stdlib`
`fscanf(f,...)`	int	read data from a file	`stdio`
`fseek(f,l,i)`	int	move the pointer for file `f` a distance `l` bytes from location `i`	`stdio`

Table 1 *Library functions*

Function	*Type*	*Description*	*Header*
`ftell(f)`	long int	return the current position of the pointer for file `f`	`stdio`
`fwrite(s,i1,i2,f)`	int	send from string `s` to file `f` i_2 data items, each i_1 bytes long	`stdio`
`getc(f)`	int	read a character from a file	`stdio`
`getchar()`	int	read a character from the standard input device	`stdio`
`gets(s)`	char*	read a string from the standard input	`stdio`
`isalnum(c)`	int	test if alphanumeric, return 0 if false and nonzero otherwise	`ctype`
`isalpha(c)`	int	test if alphabetic, return 0 if false and nonzero otherwise	`ctype`
`isascii(c)`	int	test if an ASCII character, return 0 if false and nonzero otherwise	`ctype`
`iscntrl(c)`	int	test if an ASCII control character, return 0 if false and nonzero otherwise	`ctype`
`isdigit(c)`	int	test if a decimal digit, return 0 if false and nonzero otherwise	`ctype`
`isgraph(c)`	int	test if a graphic ASCII character, return 0 if false and nonzero otherwise	`ctype`
`islower(c)`	int	test if lower case, return 0 if false and nonzero otherwise	`ctype`
`isodigit(c)`	int	test if an octal digit, return 0 if false and nonzero otherwise	`ctype`
`isprint(c)`	int	test if a printing ASCII character (hex 0x20–0x7e; octal 040–176), return 0 if false and nonzero otherwise	`ctype`
`ispunct(c)`	int	test if a punctuation character, return 0 if false and nonzero otherwise	`ctype`
`isspace(c)`	int	test if a white space character, return 0 if false and nonzero otherwise	`ctype`
`isupper(c)`	int	test if upper case, return 0 if false and nonzero otherwise	`ctype`
`isxdigit(c)`	int	test if a hexadecimal digit, return 0 if false and nonzero otherwise	`ctype`
`labs(l)`	long int	the absolute value	`math`
`log(d)`	double	the natural logarithm	`math`
`log10(d)`	double	the base-10 logarithm	`math`
`malloc(u)`	void*	allocate `u` bytes of memory, return a pointer to the beginning	`malloc`, `stdlib`
`pow(d1,d2)`	double	${d_1}^{d_2}$	`math`
`printf(...)`	int	write to the standard output device	`stdio`
`putc(c,f)`	int	write a character `c` to a file `f`	`stdio`
`putchar(c)`	int	write a character to the standard output	`stdio`
`puts(s)`	int	write a string `s` to the standard output	`stdio`
`rand()`	int	return a random positive integer	`stdlib`
`rewind(f)`	void	move the pointer to the beginning of a file	`stdio`
`scanf(...)`	int	read from the standard input	`stdio`
`sin(d)`	double	sine	`math`
`sinh(d)`	double	hyperbolic sine	`math`
`sqrt(d)`	double	square root	`math`

Table 1(continued) *Library functions*

Function	*Type*	*Description*	*Header*
`srand(u)`	void	initialise the random number generator	`stdlib`
`strcmp(s1,s2)`	int	compare two strings lexicographically, return 0 if s_1 and s_2 are identical, a positive value if $s_1 > s_2$ and a negative value if $s_1 < s_2$	`string`
`strcmpi(s1,s2)`	int	compare two strings, without considering cases, return 0 if s_1 and s_2 are identical, a positive value if $s_1 > s_2$ and a negative value if $s_1 < s_2$	`string`
`strcpy(s1,s2)`	char*	copy string s_2 to s_1	`string`
`strlen(s)`	int	return the number of characters in a string	`string`
`strset(s,c)`	char*	set all characters within string `s` to `c`, with the exception for the terminating null character \0	`string`
`system(s)`	int	pass command `s` to the operating system, return 0 if its execution is successful and a nonzero typically -1 otherwise	`stdlib`
`tan(d)`	double	tangent	`math`
`tanh(d)`	double	hyperbolic tangent	`math`
`time(p)`	long int	return the number in seconds elapsed beyond a base time	`time`
`toascii(c)`	int	convert to ASCII	`ctype`
`tolower(c)`	int	convert to lower case	`ctype`, `stdlib`
`toupper(c)`	int	convert to upper case	`ctype`, `stdlib`

Table 1(continued) *Library functions*

Emacs and GCC

Emacs is a GNU text editing programme while GCC is the GNU C Compiler. Both are considerably efficient and powerful. Compiling a C source code with GCC is usually done running a compilation script `makefile`, which can be written up to suit the various needs of the different programmers. An example of a `makefile` is given in Example 1.

Example 1. (A makefile)

```
1 # makefile, Kit Tyabandha, 13/4/05
2 cc=gcc
3 cflags=-c -g -Wall
4 lflags=-g
5 libs=-ldl -lm
6 tst : tst.o
7 ${cc} ${lflags} $< -o tst ${libs}
8 %.o : %.c
9 ${cc} ${cflags} $<
```

The `makefile` is run by issuing the command `make` followed by the name, without the extension, of the file containing the code.

The programme given in Example 2 asks for the year and then gives the date of Easter for that year.

Example 2. (Easter day)

```
1  /* Finding the date of Easter, Kit Tyabandha, 12 Dec 06 */
2  #include<stdio.h>
3  #include<stdlib.h>
4  int main(){
5    int a, b, c, d, e, day, f, g, h, i, k, l, m, n, p, tmp, year;
6    printf("\n Easter day\n Enter the year: "); scanf("%d", &year);
7    a =year%19;
8    b =year/100;
9    c =year%100;
10   d =b/4;
11   e =b%4;
12   tmp =b+8;
13   f =tmp/25;
14   tmp =b-f+1;
15   g =tmp/3;
16   tmp =19*a+b-d-g+15;
17   h =tmp%30;
18   i =c/4;
19   k =c%4;
20   tmp =32+2*e+2*i-h-k;
21   l =tmp%7;
22   tmp =a+11*h+22*l;
23   m =tmp/451;
24   tmp =h+l-7*m+114;
25   n =tmp/31;
26   p =tmp%31;
27   day =p+1;
28   printf("\n Easter for AD %d is %d ", year, day);
29   switch(n){
30   case 3: printf("March\n\n");
31   case 4: printf("April\n\n");
32   }
33   exit(0);
34 }
```

```
kit@nebula:~/prog/c$ tst
 Easter day
 Enter the year: 2007
 Easter for AD 2007 is 8 April
```

Figure 1 *Output of Example 2*

Bibliography

Byron Gottfried. *Programming with C.* Schaum's Outlines Series, McGraw-Hill, 1996

Peter Duffett Smith. *Practical astronomy with your calculator.* 2nd ed., Cambridge University Press, 1979